Rohit Raibagkar
gd4139

Comp. Networks and Programming

ECE 5650
Winter 2018

Report of Programming Assignment 2

College of Engineering

Declaration: The project and the report are my own work
I contributed 100% of my own project.
Date: 03/19/2018

Table of Contents

1. Source Code

1.1 Server Code:

```
# Library import area. All the imported libraries mentioned below
#####################################################

from builtins import *
import re, time, argparse, select, sys
from  threading import Thread
import  socket


hostName = socket.gethostname() # here, system host name is stored in variable.
hostIP = socket.gethostbyname(socket.getfqdn()) # here ip address of server is stored in variable

# Defining Sockets...
from socket import *


udpPort = 8000
tcpPort = 8001

parser = argparse.ArgumentParser(description='Networking class and app')

#def parameters( parser ):

parser.add_argument('i', type=int, help="Port Number")
parser.add_argument('j', type=int, help="Port number")

inputArgs = parser.parse_args()

if __name__ == '__main__':

    tcpPort = inputArgs.i
    udpPort = inputArgs.j



#parameters(parser)


#udpPort = 8001  # hard coded for now, use argparse later
#tcpPort = 8000  # hard coded for now, use argparse later

imRegServer = socket(AF_INET, SOCK_DGRAM)    # UDP Socket for registration purposes
imChatServer = socket(AF_INET, SOCK_STREAM) # TCP Socket for communiation purposes

imRegServer.bind(('', udpPort))  # here 8001 is input argument for UDP Port...
imChatServer.bind(('', tcpPort)) # here 8000 is input argument for TCP Port...

imChatServer.listen(2)

# Socket defining is done...

# below is the information stored received from the users...

userID = []          # List of user IDs
userAddr = []
clientIP = []        # List of client's Ips
clientTCPPort = []   # List of client's TCP ports
clientUDPPort = []   # List of client's UDP ports
clientHostName = [] # List of client's host names

# Done with storing user's information...
```

```
userCount = 0   # the variable to store number of users registered on the server.
finResponseCount = 0

while True:

    print('Server is listening at IP:\t', hostIP, '\tPort number:\t', udpPort)

    userInfo, userAddress = imRegServer.recvfrom(1024)

    print((userInfo.decode()))

    if userInfo.decode() == 'Me too ready':

        finResponseCount += 1

    if finResponseCount == 2: break


    if userInfo is not None:

        if userCount < 2:

            userInfo = re.split(r'\t+', userInfo.decode())
            userID.append(userInfo[0])
            clientIP.append(userInfo[1])
            clientUDPPort.append(int(userInfo[2]))
            clientTCPPort.append(int(userInfo[3]))
            clientHostName.append(userInfo[4])
            userAddr.append(userAddress)
            #print(userID, clientIP, clientUDPPort, clientTCPPort, clientHostName)
            imRegServer.sendto('Registration info. received'.encode(), userAddress)
            userCount += 1
    # Breking the while loop

    if userCount == 2:

        #print('Two users registered.')
        #time.sleep(2)
        imRegServer.sendto('I am ready'.encode(), userAddr[0])
        imRegServer.sendto('I am ready'.encode(), userAddr[1])
        #break

def connectionAcceptor():
    """Acepts incoming chat client connections"""

    while True:
        user, userAddress = imChatServer.accept()
        print("%s:%s joined the chat room..." %userAddress)
        user.send(bytes("Welcome to the chat room. Enter your name.", "utf8"))
        userAddrArray[user] = userAddress
        Thread(target= clientHandler, args=(user,)).start()


def clientHandler (user):
    """This function handles the incoming client connections"""

    userName = user.recv(1024).decode("utf8")
    response = 'Welcome to chat room %s. If you wish to quit, type {exit}' % userName
    user.send(bytes(response, "utf8"))
    message = "%s joined the chat room." % userName
    transmitt(bytes(message, "utf8"))
    arrayOfUsers [user] = userName

    while True:

        try:
```

```
            message = user.recv(1024)

            if message != bytes("{exit}", "utf8"):
                transmitt(message, userName + ": ")
                print(userName, message.decode("utf8"))

            else:
                user.send(bytes("{exit}", "utf8"))
                user.close()
                del arrayOfUsers[user]
                transmitt(bytes("%s left the room." % userName, "utf8"))
                break

        except:
            print('Unable to connect. Please check clients and server connections')
            break


def transmitt (message, frame = ""):

    # do nothing
    for numSockets in arrayOfUsers:

        numSockets.send(bytes(frame, "utf8") + message)


arrayOfUsers = {}
userAddrArray = {}

if __name__ == "__main__":

    imChatServer.listen(2)
    print('Server is ready to be connected.')
    trialThread = Thread(target= connectionAcceptor)
    trialThread.start()
    trialThread.join()
    imChatServer.close()
```

1.2 Client Code

```
# Library import area. All the imported libraries mentioned below
#####################################################

from builtins import *
import re, time, argparse, tkinter
import  socket
from  threading import Thread

hostName = socket.gethostname() # this command takes the host name of the Client 1 System.
hostIP = socket.gethostbyname(socket.getfqdn()) # this command takes the IP address of client 1
System...

# defining Sockets...

from socket import *

clientUDPPort = 7000    # This is UDP port of client 1, to be entered by argparse...
clientTCPPort = 7001    # This is TCP port of client 1, to be entered by argparse...

serverIP = ''  # this is ipaddress of Server, to be entered by argparse...

servUDPPort = 8000       # This is UDP port of the server, to be entered by argparse...
servTCPPort = 8001       # This is TCP port of the server, to be entered by argparse...

parser = argparse.ArgumentParser(description='Networking class and app')

#def parameters( parser ):

parser.add_argument('i', type=int, help=" Client TCP Port Number")
parser.add_argument('j', type=int, help=" Client UDP Port number")
parser.add_argument('k', type=str, help="Server IP Address")
parser.add_argument('l', type=int, help=" Server TCP Port number")
parser.add_argument('m', type=int, help=" Server UDP Port number")

inputArgs = parser.parse_args()
if __name__ == '__main__':
    clientUDPPort = inputArgs.j
    clientTCPPort = inputArgs.i
    serverIP = inputArgs.k
    servUDPPort = inputArgs.m
    servTCPPort = inputArgs.l

#parameters(parser)

userName = input('Enter User Name:')

clientData = userName + '\t' + hostIP + '\t' + str(clientUDPPort) + '\t' + str(clientTCPPort) +
'\t' + str(hostName)
readyMsg = 'Me too ready'

tcpSocket = socket(AF_INET, SOCK_STREAM)     # socket for tcp communication...
udpSocket = socket(AF_INET, SOCK_DGRAM)      # socket for udp communication...

udpSocket.sendto(clientData.encode(),(serverIP, servUDPPort))

while True:

    #udpSocket.sendto(clientData.encode(),(serverIP, servUDPPort))

    Response, serverAddress = udpSocket.recvfrom(1024)
    print(Response.decode())

    if Response.decode() == 'I am ready':
```

```
        break

udpSocket.sendto(readyMsg.encode(), (serverIP, servUDPPort))


def messageReceiver():
    """This function receives the messages from the server and other client..."""

    while True:
        # do nothing
        try:
            message = tcpSocket.recv(1024).decode("utf8")
            messagesArray.insert(tkinter.END, message)

        except OSError:
            break

def msgTransmitter (event = None):

    message = messageInput.get()
    messageInput.set("")
    tcpSocket.send(bytes(message, "utf8"))

    if message == "{exit}":
        tcpSocket.close()
        guiWindow.quit()

def closeGUIWindow ( event = None ):

    messageInput.set("{exit}")
    msgTransmitter()

guiWindow = tkinter.Tk()
guiWindow.title("FaceApp Chwitter")

messageFrame = tkinter.Frame(guiWindow)
messageInput = tkinter.StringVar()
messageInput.set("iMessage Text Message")
windowSlider = tkinter.Scrollbar(messageFrame)
messagesArray = tkinter.Listbox(messageFrame, height = 20, width = 75, yscrollcommand =
windowSlider.set)
windowSlider.pack(side = tkinter.RIGHT, fill = tkinter.Y)
messagesArray.pack(side = tkinter.LEFT, fill = tkinter.BOTH)
messagesArray.pack()
messageFrame.pack()

messageEnter = tkinter.Entry(guiWindow, textvariable = messageInput)
messageEnter.bind("<Return>", msgTransmitter)
messageEnter.pack()
messageTransmit = tkinter.Button(guiWindow, text = "Send Message", command = msgTransmitter)
messageTransmit.pack()
guiWindow.protocol("WM_DELETE_WINDOW", closeGUIWindow)


tcpSocket.connect((serverIP, servTCPPort))
trialThread = Thread(target=messageReceiver)
trialThread.start()
tkinter.mainloop()
```

2. Server Output

```
 C:\Users\rohit\Documents\Python\progAssn2\venv\Scripts\python.exe
C:/Users/rohit/Documents/Python/progAssn2/imServer.py
Server is listening at IP:    192.168.56.1    Port number:      8001
rohit 192.168.56.1     7001  7000  Rohit
Server is listening at IP:    192.168.56.1    Port number:      8001
gd4139     192.168.56.1     7003  7002  Rohit
Server is listening at IP:    192.168.56.1    Port number:      8001
Me too ready
Server is listening at IP:    192.168.56.1    Port number:      8001
Me too ready
Server is ready to be connected.
192.168.56.1:60266 joined the chat room...
192.168.56.1:60267 joined the chat room...
Rohit Hello gd4139
gd4139 Hello Rohit
Rohit How's weather in Detroit
gd4139 It's suuny here. Enjoying spring.
```

3. Client 1: Rohit Output

```
C:\Users\rohit\Documents\Python\progAssn2\venv\Scripts\python.exe
C:/Users/rohit/Documents/Python/progAssn2/imClient_1.py

Enter User Name:rohit

Registration info. received

I am ready


Chat Output >>>
```

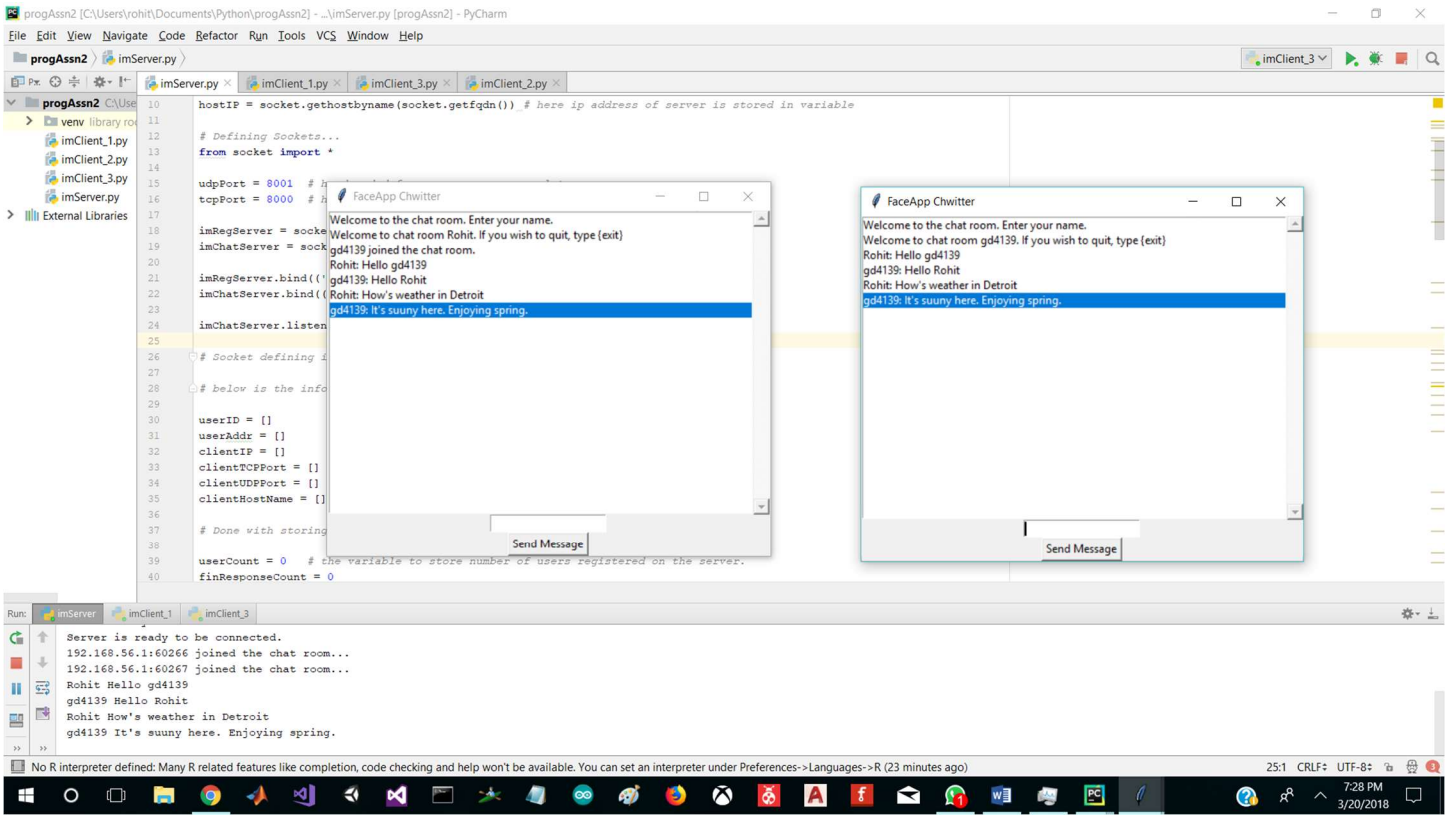
Welcome to the chat room. Enter your name.
Welcome to chat room Rohit. If you wish to quit, type {exit}
gd4139 joined the chat room.
Rohit: Hello gd4139
gd4139: Hello Rohit
Rohit: How's weather in Detroit
gd4139: It's suuny here. Enjoying spring.

4. Client 2: gd4139 Output

```
C:\Users\rohit\Documents\Python\progAssn2\venv\Scripts\python.exe
C:/Users/rohit/Documents/Python/progAssn2/imClient_3.py

Enter User Name:gd4139

Registration info. received

I am ready


Chat Output >>>
```

Welcome to the chat room. Enter your name.
Welcome to chat room gd4139. If you wish to quit, type {exit}
Rohit: Hello gd4139
gd4139: Hello Rohit
Rohit: How's weather in Detroit
gd4139: It's suuny here. Enjoying spring.

5. GUI Output


progAssn2 [C:\Users\rohit\Documents\Python\progAssn2] - ...\imServer.py [progAssn2] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
progAssn2
imServer.py
imClient_3
imServer.py
imClient_1.py
imClient_3.py
imClient_2.py
progAssn2
venv library root
imClient_1.py
imClient_2.py
imClient_3.py
imServer.py
External Libraries
hostIP = socket.gethostbyname(socket.getfqdn()) # here ip address of server is stored in variable
# Defining Sockets...
from socket import *
udpPort = 8001
tcpPort = 8000
imRegServer = socke
imChatServer = sock
imRegServer.bind(('
imChatServer.bind((
imChatServer.listen
# Socket defining i
# below is the info
userID = []
userAddr = []
clientIP = []
clientTCPPort = []
clientUDPPort = []
clientHostName = []
# Done with storing
userCount = 0 # the variable to store number of users registered on the server.
finResponseCount = 0
FaceApp Chwitter
Welcome to the chat room. Enter your name.
Welcome to chat room Rohit. If you wish to quit, type {exit}
gd4139 joined the chat room.
Rohit: Hello gd4139
gd4139: Hello Rohit
Rohit: How's weather in Detroit
gd4139: It's suuny here. Enjoying spring.
Send Message
FaceApp Chwitter
Welcome to the chat room. Enter your name.
Welcome to chat room gd4139. If you wish to quit, type {exit}
Rohit: Hello gd4139
gd4139: Hello Rohit
Rohit: How's weather in Detroit
gd4139: It's suuny here. Enjoying spring.
Send Message
Run: imServer imClient_1 imClient_3
Server is ready to be connected.
192.168.56.1:60266 joined the chat room...
192.168.56.1:60267 joined the chat room...
Rohit Hello gd4139
gd4139 Hello Rohit
Rohit How's weather in Detroit
gd4139 It's suuny here. Enjoying spring.
No R interpreter defined: Many R related features like completion, code checking and help won't be available. You can set an interpreter under Preferences->Languages->R (23 minutes ago)
25:1 CRLF UTF-8
7:28 PM
3/20/2018